CallRest

Last updated by | Łukasz Bott | 6 lut 2024 at 14:19 CET

CallRest

1. Wstęp

Funkcja CallRest umożliwia komunikację z zewnętrznymi serwisami za pomocą REST API.
(Wymagana jest podstawowa wiedza np. protokołu HTTP)

Składnia

> CallRest(<nazwa serwisu>, <nazwa metody> [, argument0, argument1, ... , argumentN])

Wymagania stawiane przez Rest API

- Obsługa typów metod GET, POST (rzadziej PUT, DELETE)
- Zwracanie danych w notacji JSON, XML (CallRest zwraca JSON'a lub XML'a)

Rekonesans Rest API

Aby pracować z REST API jakiegoś dostawcy warto przeprowadzić krótki rekonesans, aby zorientować się w jaki sposób następuje uwierzytelnianie, jakie są dostępne endpoint'y itp. Poniżej kilka sugestii

- Przegląd Swagger'a np. https://petstore.swagger.io/
- Pozyskanie danych do uwierzytelniania
- Sprawdzenie API za pomocą Postmana

2. Parametry dodawane dla każdego serwisu (jeden zestaw dla jednego serwisu)

Nazwa parametru	Przykład	Uwagi
external.rest.<Nazwa serwisu>	external.rest.TestAPI	jako wartość podawana jest lokalizacja serwisu np. https://myrestservice.astrafox.com albo http://localhost:57659/
external.rest.<Nazwa serwisu>.Login	external.rest.TestAPI.Login	wymagane tylko przy autoryzacji typu BASIC i TOKEN
external.rest.<Nazwa serwisu>.Password	external.rest.TestAPI.Password	wymagane tylko przy autoryzacji typu BASIC
external.rest.<Nazwa serwisu>.APIToken	external.rest.TestAPI.APIToken	wymagany przy autoryzacji typu TOKEN, BEARER lub CUSTOM
external.rest.<Nazwa serwisu>.AuthorizationType	external.rest.TestAPI. AuthorizationType	dostępnych jest 5 typów uwierzytelniania: **BASIC**, **TOKEN**, **BEARER**, **NOAUTH** i **CUSTOM**. Tryb CUSTOM powinien być używany jedynie gdy nie da się wykorzystać pozostałych trybów. (opcję APIKEY można uzyskać poprzez CUSTOM)
external.rest.<Nazwa serwisu>. AuthorizationHeaderParam	external.rest.TestAPI. AuthorizationHeaderParam	definiuje nazwę nagłówka odpowiedzialnego za uwierzytelnienie, wymagany jedynie przy wyborze typu uwierzytelnienia CUSTOM

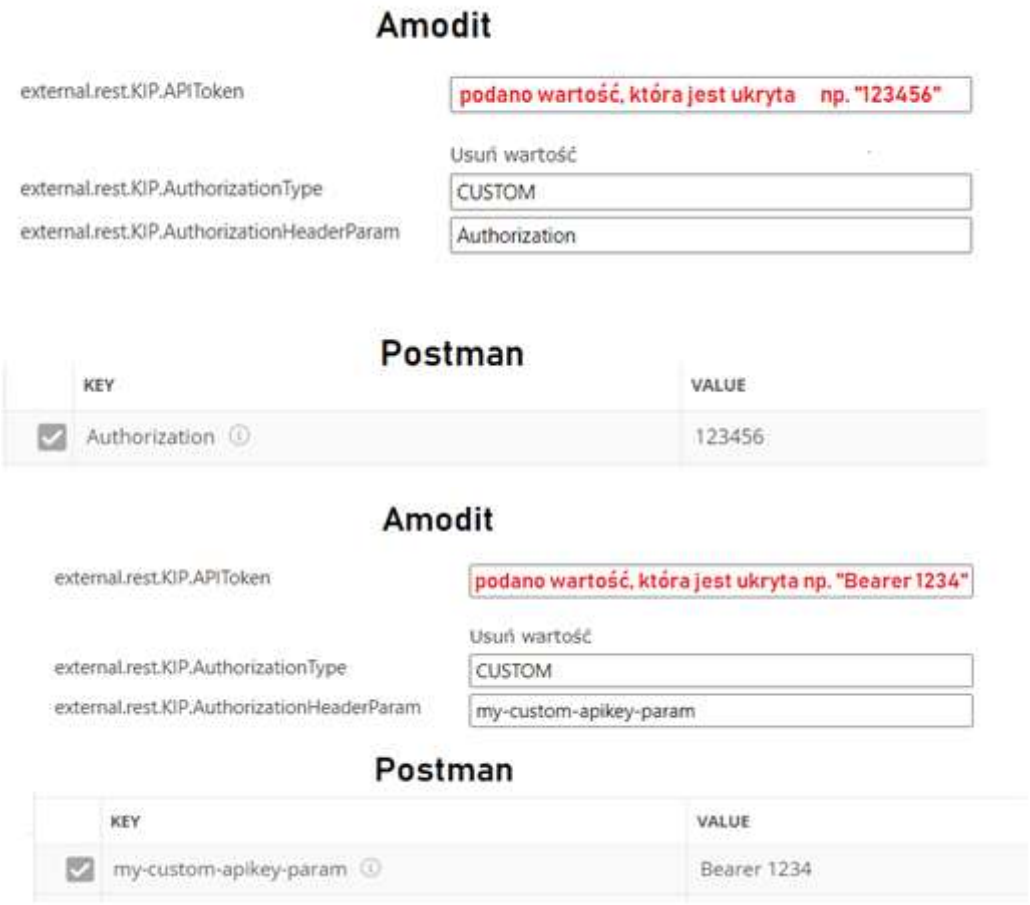
3. Parametry dodawane dla każdego endpointa w danym serwisie

Nazwa parametru	Przykład	Uwagi
external.rest.<Nazwa serwisu>.<Nazwa endpointa>.Endpoint	external.rest.TestAPI. GetSomeData.Endpoint	ścieżka wskazująca na endpoint, może zawierać query string np. /rest/api/1/issue/{{0}} lub /rest/api/issue?id={{0}}&param={{1}}
external.rest.<Nazwa serwisu>.<Nazwa endpointa>.RequestType	external.rest.TestAPI. GetSomeData.RequestType	GET, POST, PUT lub DELETE
external.rest.<Nazwa serwisu>.<Nazwa endpointa>.ContentType	external.rest.TestAPI. GetSomeData.ContentType	application/json (domyślna), application/xml, x-www-urlencoded, multipart/form-data (dla metody typu GET parametr może być pusty albo nie istnieć), text/xml (dla web serwisów typu SOAP)
external.rest.<Nazwa serwisu>.<Nazwa endpointa>.RequestBody	external.rest.TestAPI. GetSomeData.RequestBody	definiuje zawartość BODY, zapisane w notacji JSON lub XML w zależności od paramertu ContentType (dla metody typu GET parametr może być pusty albo nie istnieć)
external.rest.<Nazwa serwisu>.<Nazwa endpointa>.SOAPAction	external.rest.TestAPI.GetSomeData.SOAPAction	(dla web serwisów typu SOAP) URL używany wtedy, gdy API wymaga zdefiniowanego odnośnika dla każdego ze swoich endpointów

4. Uwierzytelnianie w CallRest

Wartość parametru AuthorizationType	Nazwa nagłówka odpowiadającego za uwierzytelnianie	Wartość nagłówka odpowiadającego za uwierzytelnianie
BASIC	Authorization	Basic <ToBase64({Login}; {Password})>
TOKEN	Authorization	Basic <ToBase64({Login}; {APIToken})>
BEARER	Authorization	Bearer {APIToken}
NOAUTH	–	–
CUSTOM	{AuthorizationHeaderParam}	{APIToken}

Przykład:



5. Dodawanie nowego serwisu.

Funkcja CallRest pobiera dane z parametrów systemowych z rozszerzeń AMODIT. Konfiguracja tych parametrów możliwa jest wyłącznie z poziomu bazy danych.

Parametry muszą mieć nazwy zaczynające się od external.rest.

Konwencja nazw dla parametrów

`external.rest.< Nazwa serwisu >[.< Nazwa endpointa >][.<Nazwa pomocnicza>]`

Parametry należy dodać do tabeli `parameters` w bazie danych AMODIT. W tym celu może być pomocny poniższy skrypt SQL:

```
INSERT INTO [parameters] (
        [parName]
        ,[parValue]
        ,[parDescription]
        ,[parGroup]
        ,[parIndex]
        ,[parDefaultValue]
        ,[parType]
        ,[parNewValue]
        ,[parModifiedById]
        ,[parAcceptedById]
        ,[parTab]
        ,[parSection]
        ,[parTitle])
VALUES (
        'external.rest.TestAPI.GetSomeData.Endpoint'
        ,'/rest/api/1/issue/{{0}}'
        ,''
        ,'Extension.MyAPIParameters'
        ,1
        ,''
        ,'s'
        ,''
        ,NULL
        ,NULL
        ,NULL
        ,NULL
        ,'external.rest.<nazwa serwisu>[.<nazwa endpointa>][.<nazwa pomocnicza>]')
```

Przykład 1 - endpoint GET bez autoryzacji

Wykorzystamy prosty serwis z faktami o kotach https://catfact.ninja/ udostępniający endpoint GET
Dla naszego przykładu Nazwa serwisu to **CatFact**, nazwa endpointa to **Fact**.

> (Wywołanie reguły: CallRest(**"CatFact"**, **"Fact"**)

Serwis nie wymaga autoryzacji więc jedyne co potrzebujemy to parametr RequestType. W naszym przypadku **GET**

Parametry w bazie danych:

	parId	parName	parValue	parDescription	parGroup	parIndex	parDefaultValue	parType	parNewValue	parModifiedById	parAcceptedById	parTab	parSection	parTitle
1	439	external.rest.CatFact	https://catfact.ninja		Extension.CatFacts	1		s		NULL	NULL	NULL	NULL	
2	440	external.rest.CatFact.Fact.Endpoint	/fact		Extension.CatFacts	2		s		NULL	NULL	NULL	NULL	
3	441	external.rest.CatFact.Fact.RequestType	GET		Extension.CatFacts	3		s		NULL	NULL	NULL	NULL	

Jeśli poprawnie dodamy parametry do bazy danych powinne one być widoczne w ustawieniach systemowych w zakładce ROZSZERZENIA AMODIT. Teraz można edytować wartości parametrów w aplikacji.

Wykorzystanie nowo dodanego serwisu w regułach:

> CallRest(<nazwa serwisu>, <nazwa metody> [, argument0, argument1, ... , argumentN])

```
jsonResponse = CallRest("CatFact", "Fact");
ShowMessage(1,jsonResponse);
```

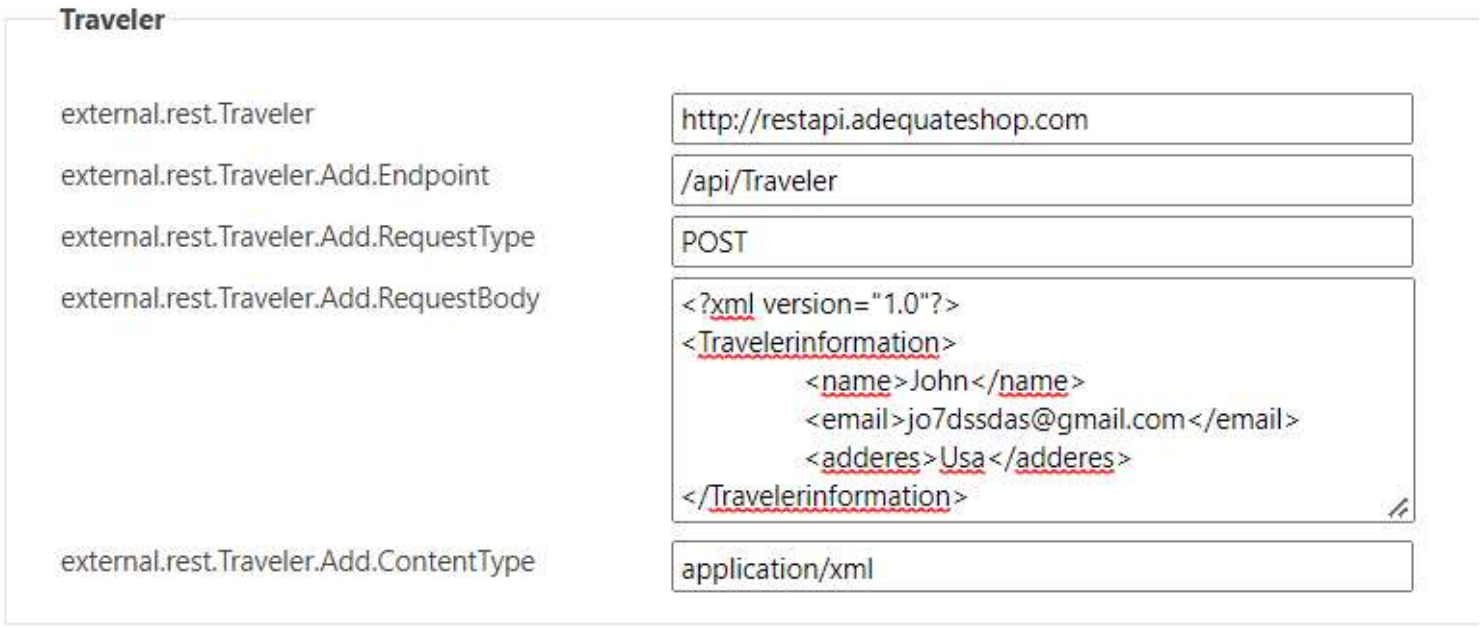
Uwaga! "CatFact" oraz "Fact" muszą mieć identyczne nazwy jak parametry ustawione w bazie danych

Przykład 2 - endpoint POST z body application/xml
https://www.appsloveworld.com/free-online-sample-xml-api-for-testing-purpose
Dla metody post musimy stworzyć parametr RequestBody oraz ContentType (domyśnie jest to application/json a chcemy użyć application/xml)

Parametry w bazie danych:

	parId	parName	parValue	parDescription	parGroup	parIndex	parDefaultValue	parType	parNewValue	parModifiedById	parAcceptedById	parTab	parSection	parTitle
1	442	external.rest.Traveler	http://restapi.adequateshop.com		Extension.Traveler	4		s		NULL	NULL	NULL	NULL	
2	443	external.rest.Traveler.Add.Endpoint	/api/Traveler		Extension.Traveler	5		s		NULL	NULL	NULL	NULL	
3	444	external.rest.Traveler.Add.RequestType	POST		Extension.Traveler	6		s		NULL	NULL	NULL	NULL	
4	445	external.rest.Traveler.Add.RequestBody	<?xml version="1.0"?><Travelerin...		Extension.Traveler	7		t		NULL	NULL	NULL	NULL	
5	448	external.rest.Traveler.Add.ContentType	application/xml		Extension.Traveler	8		s		NULL	NULL	NULL	NULL	

Oraz w ustawieniach systemowych



parType dla external.rest.Traveler.Add.RequestBody jest ustawione na "t"
Wykorzystanie:

```
xmlResponse = CallRest("Traveler","Add");
decoded = XmlEncode(xmlResponse);
ShowMessage(1,decoded);
```

Uwaga: do testowania tego endpointa trzeba zmienić mail w RequestBody (serwis nie pozwala stworzyć kolejnego wpisu z tym samym mailem - pojawi się błąd podczas wykonywania reguły)
Od wersji wydanych po 221201 można dodać jeszcze parametr SOAPAction tak aby możliwe było wywołanie web service poprzez SOAP.

6. Wsparcie dla nowych typów danych żądania (poza application/json):

- x-www-urlencoded
- multipart/form-data
- application/xml
- application/octet-stream
- text/xml *(dla web serwisów typu SOAP)*

Aby określić jaki typ danych będzie wysyłany w ramach żądania należy dodać parametr o nazwie:
`external.rest.<nazwa serwisu>.<nazwa żądania>.ContentType`
przyjmowane wartości: `application/json` (domyślna), `application/xml` , `x-www-urlencoded` , `multipart/form-data` , `text/xml`

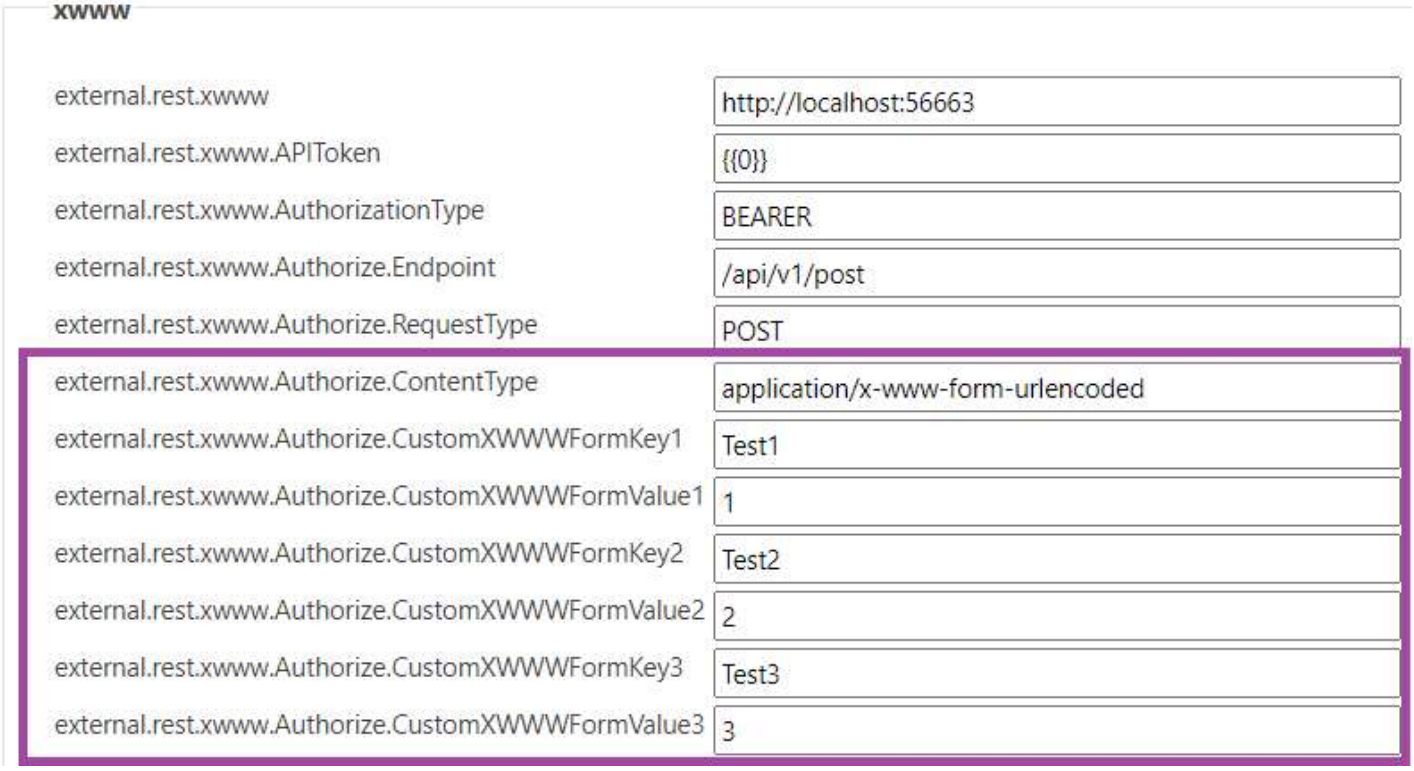
- Jeśli wybrano opcję **application/json** lub nie wybrano nic (brak parametru ContentType) to należy dodać parametr `external.rest.<nazwa serwisu>.<nazwa żądania>.RequestBody` gdzie powinien być wstawiony json wysyłany do docelowego serwisu (tutaj jest bez zmian, 99% przypadków)
- Jeśli wybrano opcję **application/xml** należy dodać parametr `external.rest.<nazwa serwisu>.<nazwa żądania>.RequestBody` (taj jak dla `application/json`)
  Przykład

```
<?xml version="1.0"?>
<Travelerinformation>
    <name>John</name>
    <email>email1@gmail.com</email>
    <adderes>Usa</adderes>
</Travelerinformation>
```

- Jeśli wybrano opcję **application/x-www-form-urlencoded** to żądanie do serwisu wysyła listę par klucz-wartość tzw. formularz. Chcąc wysłać każdą jedną taką parę parametrów należy dodać dwa parametry systemowe o nazwach wg poniższego wzoru:
  `external.rest.<nazwa serwisu>.<nazwa żądania>.CustomXWWWFormKeyN`
  `external.rest.<nazwa serwisu>.<nazwa żądania>.CustomXWWWFormValueN`

  Wartości dla parametrów stanowią nazwy kluczy i wartości wysyłane w żądaniu. **N** (dodatnia liczba naturalna) oznacza liczbę parametrów. **Musimy podawać numery par po kolei**. Jeśli mamy pary o numerach **1** i **3**, to para o numerze **3** nie zostanie wysłana wraz z żądaniem bo nie ma zdefiniowanej w parametrach systemowych pary o numerze **2**.

Przykład konfiguracji:



- Jeśli wybrano opcję **multipart/form-data** to oznacza, że będziemy przesyłać załączniki do docelowego serwisu jako listę par klucz-wartość (można też przesyłać pary klucz-wartość jak w `x-www-urlencoded`, ale na razie przesyłanie wartości tekstem nie jest wspierane). Chcąc przesłać każdy jeden załącznik trzeba stworzyć parametr z nazwą wg wzoru:

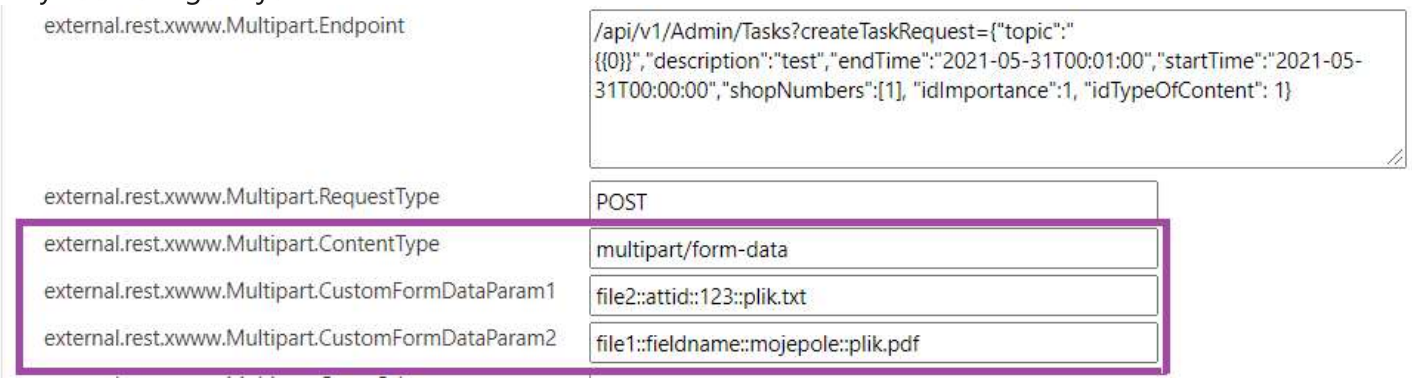
  ```
  external.rest.<nazwa serwisu>.<nazwa żądania>.CustomFormDataParamN
  ```

  Jako wartość tego parametru podajemy specjalną sekwencję wyrazów oddzieloną separatorem `::`.

  Przykład: `file1::fieldname::mojepole::plik.pdf` lub `'file2::attid::123::plik.txt`

  Dzięki takiemu zapisowi określamy jak nazywa się klucz pary klucz-wartość (załącznik), wskazujemy gdzie jest załącznik do wysłania (poprzez wskazanie id załącznika albo nazwy pola na obecnej sprawie gdzie jest załącznik) oraz nadanie nazwy pliku pod którą docelowy serwis otrzyma załącznik.

  Przykład konfiguracji:



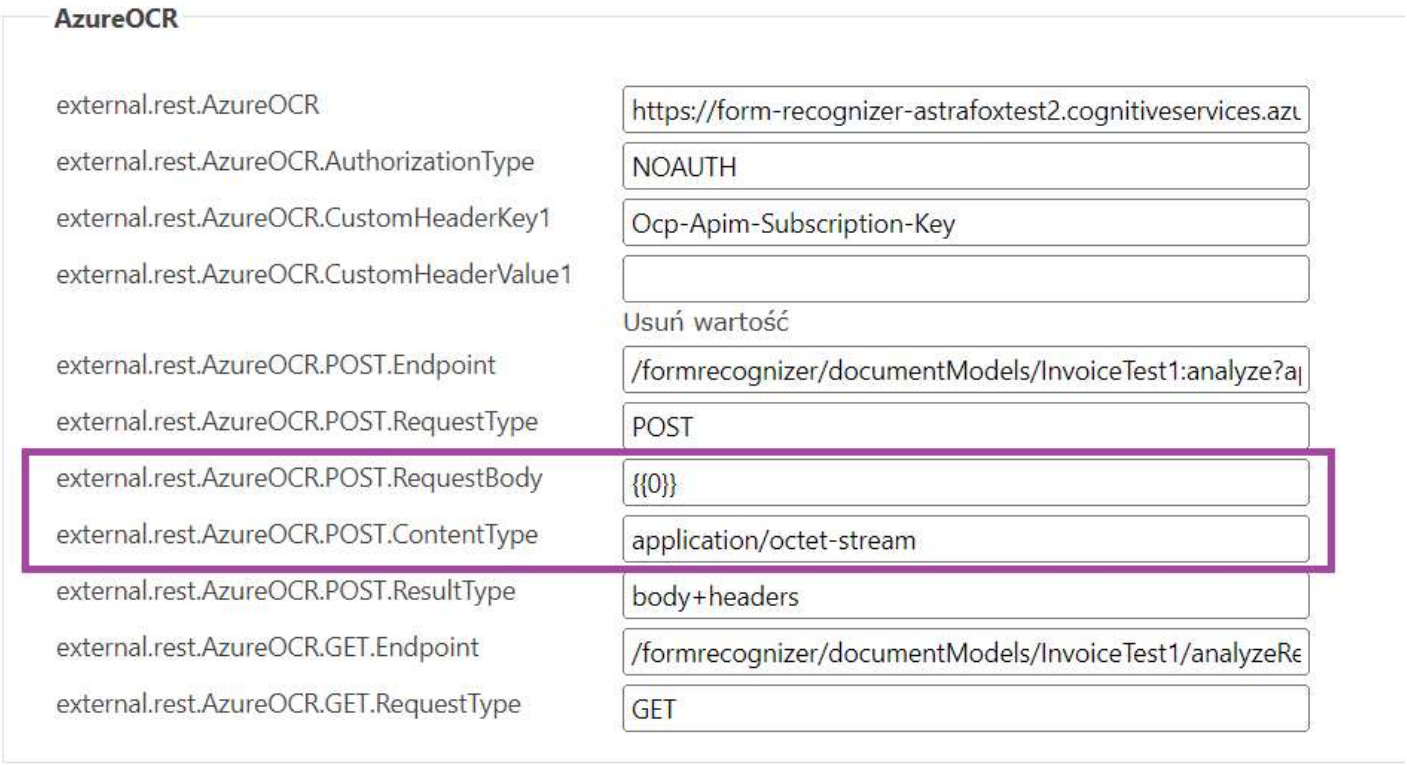
- Jeśli wybrano opcję **application/octet-stream** to oznacza to, że w bieżącym żądaniu będzie przesyłany w ramach body plik w postaci strumienia bitów. Wobec tego w ramach konfiguracji żądania w parametrach systemowych oraz opcjonalnie w treści reguły należy wskazać załącznik ze sprawy, który ma być wysyłany.

Aby wskazać załącznik należy w odpowiedni sposób wypełnić parametr systemowy `external.rest.<nazwa serwisu>.<nazwa żądania>.RequestBody`.
Jako wartość parametru można wstawiać:

- nazwę pola na formularzu w procesie (nazwa systemowa, nie wyświetlana),
- ID załącznika w systemie (można wskazać tylko załączniki znajdujące się w obrębie sprawy gdzie jest wykonywana reguła),
- nazwę pliku załącznika (można wskazać tylko załączniki znajdujące się w obrębie sprawy gdzie jest wykonywana reguła).

W sytuacjach spornych (wstawiona wartość może być zarówno np. nazwą pola oraz ID załącznika) decyduje następująca kolejność: sprawdzenie czy dokument istnieje w polu o wskazanej nazwie, potem próba wyszukania załącznika po ID i na koniec weryfikacja czy istnieje załącznik z plikiem o podanej nazwie.
Dobrą praktyką jest dynamiczne generowanie wartości w przypadku tego parametru systemowego, tj. wartością parametru jest placeholder, który za pomocą kodu reguły może być podmieniany na dowolną wartość.

Przykład konfiguracji:



7. Dynamicznie generowanie wartości w szablonie zapisanym w bazie w parametrach systemowych

7.1. Szablony parametrów systemowych oraz wstawianie wartości przez placeholder'y

Dla każdego parametru systemowego wykorzystywanego przez funkcję CallRest można wygenerować dynamicznie wartość. Zamiast podania statycznej wartości parametru można stworzyć tzw. **szablon parametru systemowego.**

Szablon parametru jest tworzony w momencie kiedy w miejscu wartości dla parametru dodatkowo wstawiamy tzw. **placeholder'y**, czyli miejsca, które podczas wywołania funkcji CallRest zostaną zastąpione wartością

reprezentowaną przez dany placeholder.

Typy placeholder'ów:

Typ placeholder'a	Przykład	Opis
Pole na sprawie	{{nazwaPola}}	W miejsce placeholder'a jest wstawiana wartość danego pola ze sprawy.
Pole słownikowe	{{nazwaPola.parametr}}	W miejsce placeholdera jest wstawiana wartość podanego parametru z wybranej w polu pozycji słownikowej. Dostępne są parametry: id, value, displayvalue, extid. Domyślnie zwracane jest id (dictposid).
Zmienna z reguły	{{_mojaZmienna}}	W miejsce placeholder'a jest wstawiana wartość wybranej zmiennej z reguły w ramach której wykonywana jest funkcja CallRest.
Argument podany do funkcji CallRest	{{0}}	Placeholder jest zastępowany przez wartość N-tego argumentu funkcji CallRest. Poza pierwszymi dwoma wymaganymi argumentami można podawać też kolejne opcjonalne argumenty. Trzeci argument funkcji CallRest (pierwszy opcjonalny) jest oznaczony jako 0. argument, czwarty argument jako 1. argument, piąty jako 2. argument, itd.

Przykład:
Sprawa posiada pole tekstowe o nazwie Miasto, w którym jest ustawiona wartość: ***Gdańsk***. Dla pól słownikowych dodatkowo można określić jaka wartość z pozycji słownika zostanie przekazana, np.: aby podać nazwę państwa a nie id należy użyć Country.value.
Ponadto sprawa posiada też regułę gdzie jest wykonywana funkcja CallRest i w danej regule istnieje zmienna o nazwie ***_populacja***. Dostępn W trakcie wykonania reguły w momencie wykonania funkcji CallRest zmienna *_populacja* ma wartość: *582205*.
Funkcja CallRest jest wywołana w następujący sposób:

```
CallRest('jakasNazwaSerwisu', 'jakasNazwaMetody', 'Pomorskie');
```

W systemie istnieje parametr systemowy utworzony pod integrację za pomocą funkcji CallRest.

Parametr (o nieistotnej nazwie) ma ustawioną wartość szablonu:

```
{
  'City': '{{Miasto}}',
  'Country': '{{Country.value}}',
  'Population': {{_populacja}},
  'Voivodeship': '{{0}}'
}
```

W momencie wykonania wspomnianej reguły CallRest w regule biznesowej dla tego samego parametru wygenerowana zostanie następująca wartość:

```
{
  'City': 'Gdańsk',
  'Country': 'Polska',
  'Population': 582205,
  'Voivodeship': 'Pomorskie'
}
```

Powyższy przykład prezentuje automatyczne wypełnienie wartości parametru na podstawie danych z kontekstu sprawy na której wykonywała się reguła biznesowa (wartość pola na formularzu sprawy) oraz kontekstu samej reguły z której wykonywana regułę biznesową (wartość zmiennej oraz dodatkowe argumenty podane przy wywołaniu funkcji CallRest).

Warto też zwrócić uwagę na to, że przy budowaniu struktur danych w notacji JSON ważna jest właściwa obsługa różnych typów wartości, które są podstawiane pod placeholder'y. Jeśli za placeholder ma być podstawiany tekst (string) to placeholder powinien być otoczony apostrofami (*single quote* lub *double quote*). Natomiast gdy placeholder będzie powodował podstawienie wartości liczbowej to nie jest to już konieczne.

7.2. Zaawansowane szablony parametrów systemowych

Zadanie: zbudować dynamicznie tekst w formacie JSON dodany jako body. Nie ma możliwości zbudować go całego sklejając stringi w kodzie reguły, bo nawiasy klamrowe wstawione w stringa nie zostaną prawidłowo sparsowane w metodzie CallRest.
Rozwiązanie: Aby nie napotkać tego problemu należy w którym z parametrów systemowych wykorzystywanych w wywołaniu metody CallRest np. RequestBody podać szkielet docelowego jsona. Jeśli chcemy wstawić listę obiektów w jsona można wykorzystać instrukcję sterującą **{{#each}}** lub instrukcje **{{#if}}** lub **{{#unless}}** jeśli chcemy warunkowo dołączać fragmenty jsona.

Wspomniane instrukcje istniały już wcześniej z wyjątkiem instrukcji **{{unless}}**, która została niedawno dodana i pełni ona rolę odwrotną względem if'a (w instrukcji if'a nie działa negacja przez ！, nie ma także wsparcia dla słowa kluczowego else).

W szablonie można też wywołać kod zgodny z językiem reguł w instrukcji **{{#code}}**. Wewnątrz może być zawarty dowolny kod zwracający jakąś wartość. Zwracana wartość będzie wpisana w podanym miejscu w wygenerowanym kodzie zgodnie z formatowaniem właściwym dla danego typu danych. Jeżeli kod zwróci string to zostanie on wpisany bez konwersji. Jeżeli kod zwróci wartość liczbową w postaci decimal to zostanie ona zapisana w formacie uniwersalnym z kropką jako separatorem dziesiętnym. Wartość datetime zostanie zapisana w formacie UTC, czyli yyyy-MM-ddTHH:mm:ss.fffzzz"

Przykład:

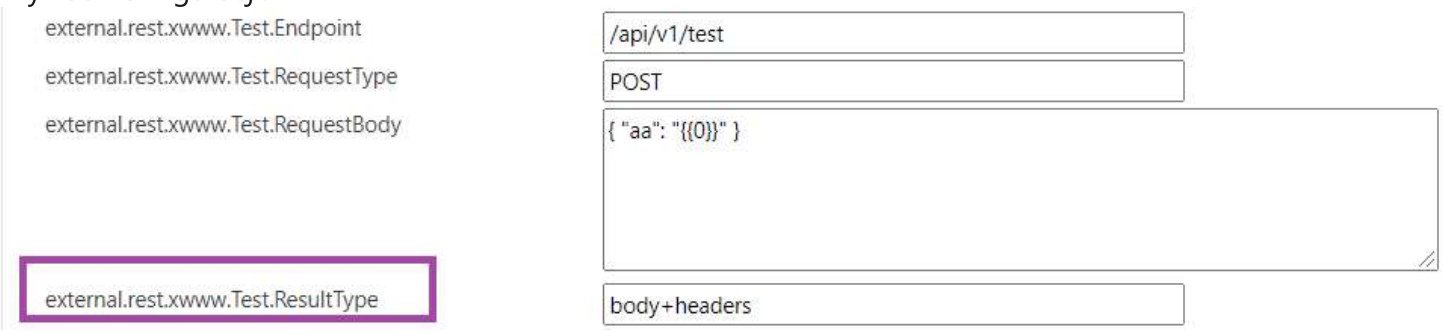
```
[
{{#each tabela}}
    {
        {{#if isOk}}
            "wartość1" : "123"
        {{/if}}

        {{#unless isOk}}
            "wartość" : "456"
        {{/unless }}

        "data":"{{#code}}ConvertTime("utc",[Date]){{/code}}"
    }
{{/each}}
]
```

8. Możliwość pobierania dodatkowych informacji z otrzymanej odpowiedzi do żądania wysłanego przez CallRest np. pobrania nagłówków odpowiedzi (gdzie mogą się znajdować np. pola z cookies)

Dla każdej z metod można stworzyć parametr
`external.rest.<nazwa serwisu>.<nazwa żądania>.RequestType`
Określa on co chcemy aby było zwrócone w regule. Mamy dostępne trzy wartości (opcje):

- **body** – wtedy zwracana jest zawartość body, opcja domyślna, która działa w przypadku braku parametru ContentType (99% przypadków użycia CallRest)
- **headers** – zwracany jest json lub xml jako obiekt o nazwie headers z listą nagłówków (klucz-wartość). Można za pomocą funkcji JsonPath albo XPath wyszukać dowolny nagłówek np. Set-Cookie.
- **body+headers** – zwracane jest zarówno body jak i nagłówki w formacie `JSON` albo `XML`. Format zwracanych odpowiedzi zależy od formatu danych odpowiedzi z docelowego serwisu.

Przykład konfiguracji:



9. Możliwość ustawiania Cookies w żądaniu CallRest

Tutaj tworzymy parametry per serwis, a nie per metoda (żądanie).

- `external.rest.<nazwa serwisu>.CustomCookieKeyN` **wymagany**
- `external.rest.<nazwa serwisu >.CustomCookieValueN` **wymagany**
- `external.rest.<nazwa serwisu>.CustomCookiePathN` **opcjonalny**

- `external.rest.<nazwa serwisu>.CustomCookieDomainN` **wymagany**
  Domain powinien zawierać adres hosta gdzie wysyłamy żądanie (bez część protokołu tj. http/https)

Każde pojedyncze ciasteczko aby było wysłane z żądaniem powinno mieć dodane do bazy i wypełnione 4 powyższe parametry. Powyższe dane można pozyskiwać np. pobierając wartość nagłówka Set-Cookies z poprzedniej odpowiedzi. Aby warunkowo wywoływać żądania nie wysyłając cookies przy tym można sobie dynamicznie ustawiać lub czyścić wartość dla parametru CustomCookieKeyN.

Przykład konfiguracji (**cookies są konfigurowane pod instancję serwisu, a nie pod konkretną metodę serwisu!**):

10. Możliwość ustawiania własnych nagłówków przy wywoływaniu żądania

W ramach funkcji CallRest możemy wysyłać żądania dokładając własne nagłówki do żądania HTTP. Taka opcja przydaje się jak klient wymaga przy uwierzytelnianiu wysyłania kilku nagłówków zamiast standardowo jednego. Należy wtedy zapisać w bazie pary parametrów klucz-wartość tworząc parametry o nazwach wg wzoru:

`external.rest.<nazwa serwisu>. CustomHeaderKeyN`

`external.rest.<nazwa serwisu>. CustomHeaderValueN`

gdzie N to kolejne dodatnie liczby naturalne.
Powyższe pary parametrów są wspólne dla wszystkich metod żądań w serwisie. Aby warunkowo wysyłać nagłówki można dynamicznie wstawiać lub czyścić wartość parametru CustomHeaderKeyN.
Przykład konfiguracji (nagłówki są konfigurowane pod serwis, a nie konkretną metodę serwisu!):